Module Thirteen

Assurance

This module describes assurance methods for guaranteeing and maintaining the security of trusted systems. The TCSEC requirements for assurance are described, and testing is discussed in detail along with other assurance methods.

Module Learning Objectives

This module presents some information that elaborates material presentedin Module 5. Most of the material presented here can be read independentlyof the other modules. Upon completion of this module, the student should:

1. Understand why assurance is necessary.
2. Understand the TCSEC assurance requirements.
3. Be familiar with the scope and philosophy of testing required for TCSEC evaluated products.
4. Be familiar with formal methods and trusted recovery and distribution.
5. Understand the documentation used to support the TCSEC assurance requirements.

Overview

Trusted systems must provide confidence that the security policy theyenforce has been correctly implemented (i.e., that the protection relevant elements of the system do, indeed, work only as intended by their designers). To accomplish this objective, two types of assurance are needed: operational assurance and life-cycle assurance.

Operational Assurance

Operational assurance applies to those features of a trusted system thatare built into it to guarantee that the system's security policy cannot be circumvented. This definition implies that the security policy must be integrated into the hardware and software protection features of the system. Operational assurance is achieved by using hardware and software to provide distinct domains and to isolate protection-critical code (System Architecture), and by using diagnostic methods to check the correctness of hardware and software operations (System Integrity). Additional operational assuranceis achieved by performing a thorough search for covert channels (Covert Channel Analysis), by distinct separating trusted user functions (TrustedF acility Management), and by ensuring that a system can recover from failurewithout compromising the security provided by the system (Trusted Recovery).

System architecture is discussed in detail in Module 7, and system integrity is sufficiently summarized in Module 4. This section describes theremaining operational assurance methods: covert channel analysis, trusted facility management, and trusted recovery.

<u>Covert Channel Analysis</u>

Covert channels are described in Module 8. Covert channel analysis attempts to identify covert channels in the system, estimate the bandwidth of each, and assess their impact on the security of the system. Although covertstorage channel analysis is required by the TCSEC at class B2 and both covert storage and timing channel analysis at class B3, any trusted system identified as possessing very high bandwidth covert channels should have these channels removed or reduced. A covert channel analysis effort for a B2 system is described in [Loepere84]. Formal methods of analysis are required at A1.

<u>Trusted Facility Management</u>

Trusted facility management consists of the administrative procedures, roles, functions (e.g., commands, programs, interfaces), privileges, and databases that are used for secure system configuration, administration andoperation. It is an area of operational assurance that is required of classes B2 and above. The objective of trusted facility management is to support security and accountability policies throughout a system's operation. To accomplish this goal, two key requirements are the separation between Administrator and Operator functions, starting at class B2, and between security-relevant and non-security-relevant functions of system administrators, starting at class B3. These separations help ensure that security-adverse effects of humanerror , misdeed, and system failure do not affect administrative functions and data.

<u>Trusted Recovery</u>

Trusted recovery provides assurance that a system that experiences an operational failure or other discontinuity will, through proceduresand/or trusted mechanisms, recover without a protection compromise. Trusted recovery is required of classes B3 and A1. Guidelines on trusted recovery are provided in [RECOV91].

Life-Cycle Assurance

Life-cycle assurance refers to the steps taken to ensure that the trustedsystem is designed, developed, and maintained using formalized and rigorouscontrols and standards, and that it has been able to withstand concerted and methodical attempts to make its security mechanisms fail (i.e., violate its security policy). Life-cycle assurance is achieved by careful evaluation and testing throughout the system design and development phases (Security Testing). Even when security testing fails to uncover any flaws in design or implementation, it cannot prove that the trusted system can never behave in some undesired manner. Therefore, additional life-cycle assurance is achieved by documenting the system's policy objectives and design features and demonstrating their implementation (Design Specification and Verification), by configuration managing security-relevant hardware and software (Configuration Management), and by controlling the distributionof the system in such a way that there are no discrepancies between the versions distributed and the master copy maintained by the vendor (Trusted Distribution).

Configuration management is described in Module 15. This sectiondescribes the remaining life-cycle assurance methods: security testing, design specification and verification, and trusted distribution.

Security Testing

Security testing is the most common technique for gaining assurance that a system operates within the constraints of a given set of policies and mechanisms. It is a method for gaining assurance about implementation, and is used to determine that the security features of a system areimplemented as designed and that they are adequate for a proposed TCSEC security class. For any trusted system, security testing must be based on the flaw hypothesis of testing, which requires that tests must be designed to demonstrate that the trusted system, in general, and the TCB, in particular, fail to meet their documented system and functional requirements. Various testing methods can be applied to security testing, as discussed in [Myers79] and [Gligor87]. Functional security testing is required starting at C1 and is intendedto uncover obvious flaws and to remove or neutralize them. At B2 and above, penetration testing is also required and is intended to demonstrate thatthe trusted system is flawless in design and nearly flawless in implementation.

Functional Security Testing

Functional security testing is required at all TCSEC classes. The strengths of functional security testing are that it can identify flaws and it is a familiar technology to developers (i.e., it is similar to testing that is normally performed). Functional security testing can be repeated for each product release to help ensure that changes do not have any unintended affects on the security mechanisms. Functional security testing is limited in that it is generally impossible to perform exhaustive testing and the generation of meaningful tests is difficult. While the testing process may handily uncover errors, the completion of this process does not provide strong assurance that there are no errors remaining. An example of a security-related functional testing effort is described in [Haley85].

There has been some debate over what components of a system require functional security testing. Basically, it is those components that are responsible for enforcing the system's security policy (i.e., the TCB). The TCB includes those components that are used by trusted users to performsecurity-relevant tasks, such as defining users, downgrading classified information, limiting access to various resources, and so on. Because such users are trusted to enter the correct information, they require either special programs restricted for their use alone or special privileges to perform such tasks, or both. As a result, such restricted programs or programs that require privilege to execute must be considered part of the TCB, and, therefore, are subject to functional security testing. Even the graphical interface used by an administrator, while not performing anything security-relevant in itself, must at least be tested for correctness if it is the sole means by which a trusted user communicates with the TCB.

A second area of debate has been over the level of system abstraction towhic h functional security testing applies. Certainly, in the development of any large and complex system, testing can occur at many different levelscorresponding to the different levels of system abstraction: module (moduletesting), system design (integration testing), system interfaces (functionaltesting), system objectives (system testing), user requirements (acceptance andinstallation testing). From a TCSEC requirements point of view, security functional testing corresponds to system and functional level testing.

Therefore, a good model for functional security testing must cover all security requirements, objectives, and functional features of a trusted system. The TCSEC provides the high-level requirements, while each evaluated system documents in its philosophy of protection what those requirements meanas specific system objectives. System objectives are also stated explicitly and implicitly throughout the system documentation. The security-relevant functional features of a trusted system are identifiable through itsinterfaces to the TCB and the system design documentation. The test plan requiredof all evaluated systems must show how the test procedures and cases of thesecurity test suite adequately cover all the security-relevant features of thetrusted system.

Penetration Testing

At B2 and above, the TCSEC requires penetration testing. This form of testing is really no different than system or functional testing in terms of itsobjectives . The purpose of security testing is to discover flaws in the trusted system, and this is the same goal for penetration testing. However, penetration testing differs from functional testing in terms of the effort made to uncoverfla ws. Penetration testing is a form of security testing that has would-bepenetrators working under no constraints other than those that are applied toordinary users attempting to circumvent the security features of an active system.The penetrators are provided with all system design and implementation documentation, including listings of system source code, manuals, and circuit diagrams. System flaws are hypothesized and attempts made to exploit the flaws. The outcome of this testing may direct changes to the system's design to eliminate the flaw or may require procedural guidelines to be added to the Trusted Facility Manual or Security Features User's Guide so as not to create conditions that permit the flaw to be exploited. The strengths and limitations of penetration testing are similar to those of security testing: flaws may be effectively found, but the penetration attempts are only as good as the penetrators and never conclusively demonstrate that the system cannot be penetrated by some means not attempted. A penetration testing effort is described in [Quann87].

Design Specification and Verification

A design specification describes system behavior in terms of axioms or in terms of exceptions, error conditions, and effects. Specifications can take several different forms, such as a security policy model (discussed in Module 5), a detailed top-level specification (DTLS), or a formal top-levelspecification (FTLS). Verification is the process of comparing two levels of design specification for proper correspondence (e.g., rules of operation with axioms, security policy model with top-level specification, top-levelspecification with source code). This comparison promotes the identification ofoverlooked errors , the review of ambiguous or incomplete functionality, and an analysis of the completeness of correctness criteria. Verification is expensive, both in time and resources, and requires special expertise.

Formal verification is the process of using formal proofs todemonstrate properties of a formal specification. A formal proof is a complete andconvincing mathematical argument that presents the full logical justification foreac h proof step. Formal proofs are generally performed with the assistance of automated tools; an overview and comparison of four formal verificationtools is provided in [Cheheyl81]. An introduction to formal specificationand verification is presented in [Gasser88].

Informal verification is less rigorous than formal verificationbecause it permits the specifications to be informally stated and informal proof techniques to be used. Informal verification techniques are not well defined, and the results are dependent on correct interpretation of requirementsand specifications written in natural language, which is prone to ambiguities and omissions. Nevertheless, if care is taken in the implementation, performance of an informal verification greatly enhances an assurance effort.

Design specification and verification requirements begin at class B1 with mandates to develop an informal (or formal) security policy modeland to informally demonstrate that it is consistent with its axioms. At class B2, a formal security policy model (FSPM) must be written and informallyproven to be consistent with its axioms, and a DTLS must be written and informally shown to be an accurate description of the TCB interface. At B3, an informal convincing argument must be made that the DTLS is consistent with the FSPM. At A1, an FTLS must be written and informally shown to be anaccurate description of the TCB interface. In addition, class A1 requires a formal FTLS-to-FSPM proof and an informal FTLS-to-code mapping.

Trusted Distribution

Trusted distribution provides assurance that the TCB running at a site isthe same as the TCB that was evaluated. A system control and distribution facility must maintain the integrity of the mapping between the description ofthe current version of the TCB and the on-site current version that isrunning . Procedures must exist that assure the TCB software, firmware, and hardware updates distributed to a site are exactly as specified by the mastercopies . Trusted distribution addresses two threats: someone tampering with asystem during transportation from the vendor site to the customer site, and a system or update arriving at the customer site that is not a legitimate systemsent by the vendor. Trusted distribution is required at class A1. A trusted distribution mechanism can also be used to detect accidental or deliberate damage to the system during operations by comparing the latest distributed version tothe currently operating version. Guidelines on trusted distribution areprovided in [DISTR88].

Relevant Trusted Product Evaluation Questionnaire Questions

2.11 TESTING

C1:

1. (a) What routines are available to test the correct operation ofthe system hardware and firmware? (b) What elements of the system hardware are tested through these routines? (c) What elements of the system firmware are tested through these routines? (d) What elements of the system hardware and firmware are not tested through these routines? (e) Does the testing include boundary and anomalous conditions? (f) Is the emphasis on diagnosing and pinpointing faults or is it on ensuring the correct operation of the system hardware and firmware?
2. (a) How are the routines in the previous question invoked?(b) Who can invoke these routines? (c) Do they run under the control of the operating system or do they run in stand-alone mode?
3. (a) When can these routines be run? (b) When should these routines be run? (c) If they run automatically, when do they run (e.g., powerup, booting, rebooting)?
4. Describe the software development testing methodology. In this description, include a discussion of various testing steps such as unit, module, integration, subsystem, system testing. This discussion should include a description of test coverage criteria and test cases development methodology.
5. Provide (a) a copy of the security test plan, a brief descriptionof its contents, or an annotated outline. (b) Does the test plan include the following information: system configuration for testing, procedures to generate the TCB, procedures to bring up the system, testing schedule, test procedures, test cases, expected test results? (c) Provide a schedule for development of thesecurity test plan if such a test plan doesn't already exist.
6. (a) How thorough is the security testing? (b) Do the testcases include nominal, boundary, and anomalous values for each input? (c) What about the combinations of inputs? (d) Describe thetest coverage criteria.
7. (a) How are the test cases developed? (b) Are they based onthe concept of functional testing, structural testing, or a combination of the two?
8. What tools and techniques (automated, manual, or a combination of the two) will be used to do the functional and/or structural analysis in order to develop a thorough set of test cases?

B1:

9. How do you plan to ascertain that errors have been minimized in the system?

B2:

10. What is the role of the descriptive top-level specification (DTLS) in the functional and/or structural analysis done in order to develop a thorough set of test cases?

11. (a) Do you plan to develop scenarios for penetration testing?(b) If so, what methodologies will be used?

12. How do you plan to compute and verify the bandwidths of covert channels?

A1:

13. What is the role of the formal top-level specification (FTLS) inthe functional and/or structural analysis done in order to develop a thorough set of test cases?

2.12 MODELING AND ANALYSIS

B2:

5. (a) Provide a copy of the Verification Plan, a brief descriptionof its contents, or an annotated outline. (b) Provide a schedule for completion of the Verification Plan.

9. (a) What tools, techniques and methodologies are used to represent the descriptive top-level specification (DTLS)?(b) What portions of the TCB are represented by the DTLS?

B3:

11. What tools, techniques and methodologies are used to show that the DTLS is consistent with the formal security policy model?

A1:

12. (a) What tools, techniques and methodologies are used to represent the formal top-level specification (FTLS)? (b)What portions of the TCB are represented by the FTLS?

13. What tools, techniques and methodologies are used to verify or show that the FTLS is consistent with the formal security policy model?

14. What tools, techniques and methodologies are used to identify the implemented code modules that correspond to the FTLS?

15. What tools, techniques and methodologies are used to show that the code is correctly implemented vis-a-vis the FTLS?

2.13 OTHER ASSURANCES

B2:

8. Describe the version control or other philosophy to ensure that the object code corresponds to the correct source code, which in turn is accurately abstracted in the DTLS.

11. How do you plan to show consistency between the DTLS and the code?

A1:

15. Describe the version control or other philosophy which ensures that the FTLS continues to accurately describe the system through system changes.
16. How do you plan to show consistency among the FTLS, DTLS and the code?
17. Describe the tools, techniques and procedures used to ensure the integrity of the TCB elements (hardware, firmware, software, documents, etc.) supplied to the customers (e.g., trusted courier, electronic seals, physical seals).

Required Readings

TCSEC85 National Computer Security Center, *Department of Defense Trusted Computer Security Evaluation Criteria,* DoD 5200.28-STD, December 1985.

There are a large number of requirements related to assurance that are in the TCSEC. A number of these requirements do not appear until the higher assurance classes. Each subject area is listed below along with the Sections where its requirements appear in the TCSEC:

- System Architecture: 2.1.3.1.1, 2.2.3.1.1, 3.1.3.1.1, 3.2.3.1.1, 3.3.3.1.1, and 4.1.3.1.1 (summarized on page 105).
- System Integrity: 2.1.3.1.2, 2.2.3.1.2, 3.1.3.1.2, 3.2.3.1.2, 3.3.3.1.2, and 4.1.3.1.2 (summarized on page 106).
- Covert Channel Analysis: 3.2.3.1.3, 3.3.3.1.3, and 4.1.3.1.3 (summarized on page 97).
- Trusted Facility Management: 3.2.3.1.4, 3.3.3.1.4, and 4.1.3.1.4.
- Trusted Recovery: 3.3.3.1.5 and 4.1.3.1.5 (summarized on page 108).
- Security Testing: 2.1.3.2.1, 2.2.3.2.1, 3.1.3.2.1, 3.2.3.2.1, 3.3.3.2.1, and 4.1.3.2.1 (summarized on page 104).
- Design Specification and Verification: 3.1.3.2.2, 3.2.3.2.2, 3.3.3.2.2, and 4.1.3.2.2 (summarized on pages 98-99).
- Configuration Management: 3.2.3.2.3, 3.3.3.2.3, and 4.1.3.2.3 (summarized on pages 96-97).
- Trusted Distribution: 4.1.3.2.4 (summarized on page 106).

The assurance control objectives are described in Section 5.3.3. The policy basis for the TCSEC control objectives for assurance is

described in Section 7.5. A guideline on covert channels is provided in Section 8.0. A guideline on security testing is provided in Section 10.0.

INTERP94 National Computer Security Center, *The Interpreted TCSEC Requirements*, (quarterly).

The following Interpretation is relevant to system architecture:

C1-CI-04-85	System Architecture

The following Interpretation is relevant to system integrity:

I-0144	Availability of diagnostics

The following Interpretations are relevant to covert channel analysis:

C1-CI-02-84	Security Testing
C1-CI-07-84	Audit

The following Interpretations are relevant to security testing:

I-0170	Functional tests required for object reuse
C1-CI-01-83	Security Testing
C1-CI-02-84	Security Testing

The following Interpretations are relevant to design specification and verification:

I-0254	UNIX-style manual pages as DTLS
C1-CI-01-87	FTLS Accuracy

The following Interpretation is relevant to configuration management:

I-0285	CM comparison source or object?

None of the Interpretations are relevant to trusted facility management, trusted recovery, or trusted distribution.

Gasser88 Gasser, M., *Building a Secure Computer System*, Van Nostrand Reinhold Co., N.Y., 1988.

Section 4.3 provides an introduction to assurance and the system's development path. Section 10.2.3 provides some thoughts on assurance techniques. Chapter 12 talks about formal specification and verification. It describes formal specification techniques, properties of formal specifications, proof techniques, and how a specification to model correspondence is accomplished. The student should not be too concerned with the detailed information on formal verification.

DISTR88 National Computer Security Center, *A Guide to Understanding Trusted Distribution in Trusted Systems*, NCSC-TG-008, Version 1, 15 December 1988.

This document provides guidance on the TCSEC trusted distribution requirements. It discusses protective packaging, secure product transportation, and site validation.

Haley85 Haley, C., and Mayer, F., “Issues on the Development of Security Related Functional Tests,” *Proceedings of the 8th National Computer Security Conference*, pp. 82-85, September 1985.

This paper presents some initial efforts to describe what the TCSEC requires for security testing. It also illustrates the notion of boundary-value coverage analysis.

Loepere84 Loepere, K., *Resolving Covert Channels within a B2 Class Secure System*, Multics Development Center -- Honeywell Information Systems, 1984.

This paper discusses practical aspects of finding covert channels. While covert channels are not a direct concern of RAMP systems, this paper offers very good descriptions of the system-wide concerns a vendor would face. << Note: this paper is included in the Required Reading materials for Module 8. >>

RECOV91 National Computer Security Center, *A Guide to Understanding Trusted Recovery in Trusted Systems*, NCSC-TG-022, Version 1, 30 December 1991.

This document provides guidance on the TCSEC trusted recovery requirements. It discusses good practices for trusted recovery.

STTD93 National Computer Security Center, *A Guide to Understanding Security Testing and Test Documentation in Trusted Systems*, NCSC-TG-023, Version 1, July 1993.

This document provides an in-depth guide to security testing, emphasizing the testing of systems to meet the TCSEC requirements. It gives system developers and vendors suggestions and recommendations on how to develop testing and testing documentation that will be found acceptable by an NSA evaluation team.

Sullivan89 Sullivan, E., “What is a Trusted System Anyway?,” *Proceedings of SHARE72*, July 1989.

In answering the paper's title question, assurance and assurance techniques are summarized.

TFM89 National Computer Security Center, *A Guide to Understanding Trusted Facility Management*, NCSC-TG-015, Version 1, 18 October 1989.

“This document provides guidance to manufacturers on how to incorporate functions of trusted facility management into their systems; to system evaluators and accreditors on how to evaluate the design and implementation of trusted facility management functions; and to end users on how to use these functions

effectively, e.g., on how to avoid common pitfalls of system management."

Supplemental Readings

Cheheyl81	Cheheyl, M., Gasser, M., Huff, G., and Millen, J., "Verifying Security," *Computing Surveys*, Vol. 13, No. 3, September 1981. This paper provides an introduction to verification and a comparison of four automated specification and formal verification tools: Gypsy, HDM, FDM, and Affirm.
CM88	National Computer Security Center, *A Guide to Understanding Configuration Management in Trusted Systems*, NCSC-TG-006, Version 1, 28 March 1988. This document provides guidance on the TCSEC configuration management requirements and discusses issues involved in implementing configuration management in the development and life-cycle of a trusted system.
Kemme86b	Kemmerer, R., "A Brief Summary of a Verification Assessment Study," *Proceedings of the 9th National Computer Security Conference*, pp. 1-6, September 1986. Presents an overview of the five volume report found in [Kemme86a] that reviews the Affirm, FDM, Gypsy, and enhanced HDM verification systems in great depth.
Linde75	Linde, R., "Operating System Penetration," *Proceedings of the 1975 National Computer Conference*, pp. 361-368, 1975. An early paper that examines a penetration methodology.
Quann87	Quann, J. and Belford, P., "The Hack Attack: Increasing Computer System Awareness of Vulnerability Threats," *AIAA/ ASIS/IEEE Third Aerospace Computer Security Conference*, pp. 155-157, December 1987. This paper discusses a penetration study effort of NASA spaceflight mission support systems that used external computer hackers under controlled conditions.
VERIF89	National Computer Security Center, *Guidelines for Formal Verification Systems*, NCSC-TG-014, Version 1, 1 April 1989. This document provides guidance on the requirements and evaluation process for formal verification systems that are candidates for the NSA's Endorsed Tools List.

Other Readings

Barker89	Barker, W., "Use of Privacy-Enhanced Mail for Software Distribution," *Proceedings of the 5th Aerospace Computer Security Applications Conference*, pp. 344-347, December 1989.

Casey88	Casey, T., Vinter, S., Weber, D., Varadarajan, R., and Rosenthal, D., “A Secure Distributed Operating System,” *Proceedings of the IEEE 1988 Symposium on Security and Privacy*, pp. 27-38, April 1988.
Freema88	Freeman, J., Neely, R., and Megalo, L., “Developing Secure Systems: Issues and Solutions,” *Proceedings of the 4th Aerospace Computer Security Applications Conference*, pp. 183-190, December 1988.
Gligor87	Gligor, V., Chandersekaran, C., Jiang, W., Johri, A., Luckenbaugh, G., and Reich, L., “A New Security Testing Method and Its Application to the Secure Xenix Kernel,” *IEEE Transactions on Software Engineering*, Vol. SE-13, No. 2, pp. 169-183, February 1987.
Kemme86a	Kemmerer, R., *Verification Assessment Study Final Report*, NCSC-C3-Cr01-86, Vols. I-V, March 1986.
Lu89	Lu, M.M. and Mayer, B.A., “Guidelines for Formal Verification Systems: Overview and Rationale,” *Proceedings of the 12th National Computer Security Conference*, pp. 75-82, October 1989.
Myers79	Myers, G., *The Art of Software Testing*, John Wiley & Sons,N.Y., 1979.
Neely85	Neely, R. and Freeman, J., “Structuring Systems for Formal Verification,” *Proceedings of the 1985 IEEE Symposium on Security and Privacy*, pp. 2-13, April 1985.
Smith88	Smith, B., Reese, C., Lindsay, K., and Crane, B., “A Description of a Formal Verification and Validation (FVV) Process,” *Proceedings of the 4th Aerospace Computer Security Applications Conference*, pp. 401-408, December 1988.
Stauffer86	Stauffer, B. and Fujii, R., “Informal Verification Analysis,” *Proceedings of the 9th National Computer Security Conference*, pp. 126-129, September 1986.
Weiss88	Weiss, J. and Amoroso, E., “Ensuring Software Integrity,” *Proceedings of the 4th Aerospace Computer Security Applications Conference*, pp. 323-330, December 1988.
Young85	Young, W., Boebert, W., and Kain, R., *Proving a Computer System Secure*, reprinted in Tutorial Computer and Network Security IEEE Computer Society Order Number 756, 1985.